This application is submitted in the name of Inventors Michael D. Johas Teener, and Colin Whitby-Strevens, assignor to Apple Computer, Inc. a California Corporation.

SPECIFICATION

1394b S800 OVER CAT5 CABLING USING GIGABIT ETHERNET PHY

FIELD OF THE INVENTION

[0001] The present invention relates broadly to devices connected to a high-speed serial bus. Specifically, the present invention relates to bridging data transmission between IEEE 1394-compliant devices and Ethernet-compliant devices.

BACKGROUND

[0002] A "bus" is a collection of signals interconnecting two or more electrical devices that permits one device to transmit information to one or more other devices. There are many different types of buses used in computers and computer-related products. Examples include the Peripheral Component Interconnect ("PCI") bus, the Industry Standard Architecture ("ISA") bus and the Universal Serial Bus ("USB"), to name a few. Bus operation is usually defined by a standard that specifies various concerns such as the electrical characteristics of the bus, how data is to be transmitted over the bus, how requests for data are acknowledged, and the like. Using a bus to perform an activity, such as transmitting data, requesting data, etc., is generally called running a "cycle."

Standardizing a bus protocol helps to ensure effective communication between devices connected to the bus, even if such devices are made by different manufacturers. Any company wishing to make and sell a device to be used on a particular bus, provides that device with an interface unique to the bus to which the device will connect. Designing a device to particular bus standard ensures that device will be able to communicate properly with all other devices connected to the same bus, even if such other devices are made by different manufacturers.

[0003] Thus, for example, an internal fax/modem (i.e., internal to a personal computer) designed for operation on a PCI bus will be able to transmit and receive data to and from other devices on the PCI bus, even if each device on the PCI bus is made by a different manufacturer.

[0004] Problems occur when devices located on buses or networks using different low-level protocols are made to communicate with each other. One example involves two very popular standards, the IEEE 1394 family of serial bus protocols, and the IEEE 802.3 family of Ethernet network protocols. Despite the fact that there are versions of both protocols that use the same cables and connectors, and both support the higher level "Internet Protocol" (IP), devices implementing the Ethernet-compliant network interface are unable to communicate with devices implementing the 1394-compliant bus interface because of the differences existing between the respective protocols. Because of the large number of existing devices that use one protocol or the other, this communication gap is likely to widen as standards are developed in the two protocols. Thus, there is a heartfelt need for a solution that bridges the communication gap between protocols and effectively allows devices to communicate with each other across different bus or networking architectures.

SUMMARY

[0005] The present invention solves the problems discussed above by providing a method and apparatus for communicating data between a gigabit Ethernet architecture and a IEEE 1394-compliant architecture. In an embodiment, the present invention provides a hub device that incorporates a 1394-compliant physical layer and an Ethernet physical layer and bridges communication between the two architectures at the Internet Protocol layer.

[0006] In another embodiment, the present invention provides a method of communicating data between an Ethernet system and a 1394-compliant system by transmitting an S800 1394b stream within a 1000 BASE-T transmit stream by inserting an illegal 1394b symbolinto the stream approximately once for every 59 regular symbols transmitted in the 1000 BASE-T transmit stream. This illegal symbol is inserted to compensate for the fact that the bit rate for S800 after an 8B10B encode is 983.04 Mbps +/- 100 ppm, and the bit rate for 1000 BASE_T stream is slightly faster at 1000 Mbps +/- 50ppm. The illegal symbol can be inserted into the transmit stream at a fixed rate, or in another embodiment, by utilizing a one-symbol queue that is fed by a 1394b transmitting port and drained by logic which supplies it to the 1000 base Tx PHY. In this embodiment, when the queue underflows, the illegal symbol is generated by the logic with drains the queue. To receive a 1394-compliant stream over an Ethernet bus, the same approach can be followed, The illegal symbols received are recognized and ignored. In an embodiment, a receive FIFO is centered when the start of a packet, is recognized, i.e. enough symbols are buffered to allow for the jitter in the arrival rate resulting from deleting the illegal symbols, and also to compensate for ppm clock differences which might result in symbols being placed in the FIFO at a slower rate than they are removed.

[0007] In another aspect of the present invention, an autonegotiation process executes to select either an Ethernet protocol or a 1394b protocol to be used on a connected communication medium. In this manner, data from a gigabit Ethernet architecture, or data from a1394-compliant architecture, can be transmitted across a common medium, without any manual configuration required of a user. In this embodiment, five registers are included on a 1394b PHY, but the port in the 1394b PHY runs off the same 125MHz clock as is provided by the gigabit media independent interface (GMII). A modulo 5 counter is included on the 1394b PHY to control symbol insertion into the transmit queue located on the 1394b side. The modulo 5 counter iteratively counts from 0 to 4. If the modulo 5 counter has a zero value, then the 1394b side does not push a 10-bit number into the FIFO. For cases where the counter has a non-zero value, the 1394b side pushes the 10-bit symbol assembled to preserve clock consistency across different clock domains.

[0008] In yet another aspect, the present invention provides a method of transmitting data across a high-speed serial bus. In accordance with an IEEE 1394-compliant PHY's TX symbol clock, a 10-bit symbol is generated on an IEEE 1394-compliant PHY. The 10-bit symbol is scrambled and encoded and placed in a FIFO. In accordance with an IEEE 802.3-compliant PHY's TX clock, the 10-bit symbol is removed from the FIFO, an 8-bit byte is derived from the removed 10-bit symbol; and sent to an IEEE 802.3-compliant PHY. A symbol is removed from the FIFO on four out of every five GMII TX clock cycles. A null 10-bit symbol is placed in the FIFO if there are no 10-bit symbols present in the FIFO. The modulo 5 counter is used to derive the 8-bit byte in the following manner: First, the 8-bit byte is derived from the 10-bit symbol by using 8 bits from the extracted 10-bit symbol, and the two remaining bits are stored. The next 8-bit byte is derived by extracting from the FIFO a second 10-bit symbol and assembling an 8-bit byte from the stored two bits and six bits from the extracted second 10-bit

symbol, the four remaining bits from the extracted second symbol are stored and the second 8-bit byte is sent to the IEEE 802.3-compliant PHY. A third 8-bit byte is derived by extracting from the FIFO a third 10-bit symbol and assembling an 8-bit byte from the four stored bits and four bits from the third extracted symbol. The six remaining bits from the extracted third symbol are stored and the third 8-bit byte is sent to the IEEE 802.3-compliant PHY. A fourth 8-bit byte is derived by extracting from the FIFO a fourth 10-bit symbol, and assembling an 8-bit byte from the six stored bits and 2 bits from the extracted fourth 10-bit symbol. The eight remaining bits from the extracted fourth symbol are stored and the fourth 8-bit byte is sent to the IEEE 802.3-compliant PHY. A fifth 8-bit byte is derived from the stored eight remaining bits and sent to the IEEE 802.3-compliant PHY.

[0009] On the receiving side, the method comprises receiving an 8-bit byte, and in accordance with a GMII RX clock, if the received 8-bit byte contains a null symbol, then deleting the null symbol; else storing the 8-bit byte in a first register; receiving a second 8-bit byte that does not contain a null symbol and storing it in a second register. A 10-bit symbol is assembled from the 8-bit byte stored in the first register and appending two bits from the 8-bit byte stored in the second register. The assembled 10-bit symbol is placed in a first FIFO. In accordance with a second clock, the 10-bit symbol is removed from the first FIFO. 8B10B and control decoding is performed on it before it is placed in a second FIFO.
In accordance with a third clock, the decoded 10-bit symbol is removed from the second FIFO; and sent to an IEEE 1394-compliant PHY. The frequency of null character deletion is used to control a phased locked loop associated with the second clock.

[0010] In still another aspect, the present invention provides a method of transmitting data across a high-speed serial bus, comprising: in accordance with a first TX symbol clock: generating a 10-bit symbol on an IEEE 1394-compliant

PHY having a port interface, placing the generated 10-bit symbol on the port interface, performing flagged encoding the 10-bit symbol, and placing the 10-bit symbol in a FIFO. In accordance with a second TX clock, the 10-bit symbol is removed from the FIFO, an 8-bit byte is derived from the removed 10-bit symbol; and sent to an IEEE 802.3-compliant PHY.

[0011] In yet another aspect, the present invention provides an apparatus for transmitting data across a high-speed serial bus, the method comprising: an IEEE 802.3-compliant PHY having a GMII interface; an IEEE 1394-compliant PHY in communication with the IEEE 802.3-compliant PHY; a first connection, the first connection for transmitting data between a device and the IEEE 802.3-compliant PHY; and a second connection, the second connection for transmitting data between a device and the IEEE 1394-compliant PHY. The IEEE 1394-compliant PHY is in communication with the IEEE 802.3-compliant PHY via a switch. The switch determines whether data transmission is be routed to the IEEE 802.3-compliant PHY or the IEEE 1394-compliant PHY. An autonegotiation mechanism determines whether data is to be routed between the IEEE 802.3-compliant PHY and the IEEE 1394-compliant PHY, or whether data is to be routed through the IEEE 802.3-compliant PHY to the first connection.

[0012] Other features and advantages of the present invention will become apparent from reading the following detailed description, when considered in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIG. 1 illustrates in block diagram form structural and function elements in an embodiment;

[0014] FIG. 2 illustrates functional components included in a port in accordance with an embodiment of the present invention;

[0015] FIG. 3 illustrates in data flow diagram form translation between 8-bit bytes and 10-bit symbols in accordance with an embodiment of the present invention;

[0016] FIG. 4 illustrates the relationship between 10-bit symbols and 8-bit characters;

[0017] FIG. 5 illustrates functional components in an alternative embodiment of a port in accordance with the present invention;

[0018] FIG. 6 illustrates in data flow diagram form an embodiment of the present invention that uses 802.3 clause 40 idle bytes instead of illegal symbols to perform translation between 10-bit symbols and 8-bit characters but does so without having to recover an RX receive clock; and

[0019] FIG. 7 illustrates in data flow diagram form an embodiment of the present invention that uses 802.3 clause 40 idle bytes instead of illegal symbols to perform translation between 10-bit symbols and 8-bit characters but does so without having to recover a TX symbol clock.

SUMMARY OF THE INVENTION

[0020] Various embodiments of the present invention utilize a 1000 BASE-T PHY that transmits and receives at no less than 0.9999 gbit/sec, and a 1394b PHY having a cable interface that utilizes a bit stream rate of no more than

0.98304+100ppm=0.9831383 gbit/sec. Thus, the fastest S800 1394b transmission rate can be accommodated by the slowest 1000 BASE-T stream.

[0021] In an embodiment, the present invention upshifts an S800 1394b stream by inserting an illegal symbol within the transmit stream approximately once for every 59 regular symbols. In an embodiment, the illegal symbol is either all ones or all zeros since those are the furthest Hamming distance from 1394b control symbols or data symbols. In various embodiments, insertion of this illegal symbol can be performed at a fixed rate, or by utilizing a one-symbol FIFO that is fed by the 1394b transmitting port and drained by the 1000 base TX PHY. In this embodiment, when the FIFO underflows, the extra symbol is generated. On receive, the same approach can be followed. In an embodiment, a FIFO is centered at the start of a packet.

[0022] In another embodiment of the present invention, the 1000 base-T transmit stream requires a number of Ethernet IDLE symbols to appear periodically. A small transmit FIFO is provided and, when it underflows, transmission of the 1394b symbols is halted and the 1000 base TX PHY is instructed to transmit Ethernet IDLE symbols for 11 Ethernet symbol times. During this time, the transmit FIFO is filling. After the 11 Ethernet symbol times has passed, transmission of 1394b symbols is resumed. Insertion of these idle symbols satisfies a requirement for 1000 base T transmission.

[0023] Directing attention to FIG. 1, a functional representation of an embodiment of the present invention is provided. The functional elements illustrated in FIG. 1 can be incorporated into a node device that communicates over a serial bus as described above to allow a user to implement a variety of Internet-based functions over a 1394-compliant bus architecture. Integrated PHY chip 100 includes 802.3 clause 40 PHY 102 having external connection 103 and

GMII interface 104, and integrated 1394 PHY106, having at least one external connector 107, controllably connected to 802.3 Clause 40 PHY 102. 1394 PHY 106 includes at least one 1394 port 108, each having a FIFO 110 for delivering bits, characters, or symbols to 1394 arbitration and data repeat module 112. Also included in 1394 PHY 106 is port 114, which also communicates bits, characters or symbols to 1394 arbitration and data repeat module 112 via FIFO 116. In an embodiment, FIFO 116 is sized larger than FIFO 110 to accommodate additional latencies experienced on port 114. Port 114 also connects to 802.3 Clause 40 PHY 102 via a switch 118. In an embodiment, switch 118 is controllably set between port 114 and GMII interface 104 according to an autonegotiation process described herein. 1394 arbitration and data repeat module 112 performs bus arbitration as well as routing data received between link layer interface 122 and/or ports 108, 114. Arbitration is handled by module 112 to determine whether PHY 106 is to transmit over a connected serial bus or receive data over the serial bus. Link layer 124 assembles a stream of symbols received over link layer interface 122 into packets that are transmitted forward over protocol layer 126. Conversely, link layer 124 translates packets received from protocol layer 126 into a stream of symbols that can be passed to the link layer interface 122. In an embodiment, link layer 124 and protocol layer 126 are implemented on a shared chip 128. IP 1394 firmware module 130 includes logic for implementing application-specific protocols that transport Internet protocol (IP)-format packets over 1394 hardware and logic. IP bridge firmware module 132 bridges packets transmitted between IP bridge firmware module 132 and implements a separate IP subnet or an 802.3 subnet and bridges communication between these two types of subnets according to IP standards. Alternatively, IP bridge firmware module 132 manages a common subnet between 802.3 Clause 40 PHY 102 and 1394 PHY 106 and ensures that IP addressing of transmitted packets is distinct. The GMII side of the present invention includes 802.3 IP firmware module 134, that performs similar functions for 802.3 Claude 40 PHY 102 as IP 1394 firmware module 130 provides

for 1394 PHY 106, and chip 138, containing GMII side 802.3 protocol layer module 140 and 802.3 media access card (MAC) 142, which perform similar functions for the GMII side as 1392 protocol layer 126 and 1394 link layer module 124 perform for the 1394 side, respectively.

[0024] Directing attention to FIG. 2, port 114 can be conceptualized has having a 1394 arbitration side of functionality and a GMII side of functionality. In an embodiment, FIFO 200 is included on port 114, and in combination with register 202 and register 204, and modulo 5 counter 206 is used to extract 10-bit symbols from the 1394 side of port 114 and process portions of the extracted symbols into 8-bit characters to be transmitted to GMII interface 104, thus bridging communication between a 1394-compliant system and a gigabit Ethernet systems. In an embodiment, registers 208 and 210 are provided for assembly of 10-bit symbols from 8-bit characters arriving from GMII interface 104 over connection 120. However, in another embodiment, these 10-bit symbols can be assembled using registers 202 and 204.

[0025] Directing attention to FIG. 3, autonegotiation selects whether an Ethernet or a protocol from the 1394 family of protocols is to be used on the connection thus determining the selection made by switch 118. In this embodiment, the 1394 side runs off the same 125MHz clock as is provided by GMII interface 104. The 1394 side of port 114 generates 10-bit symbols using a 98.304 MHz clock and pushes them into FIFO 200 using this clock. At the other end of FIFO 200, TX Adaptation module 220 extracts the 10-bit symbols from FIFO 200 using the GMII 125 MHz clock and modulo 5 counter 206. Modulo 5 counter 206 supplies values ranging from 0 to 4. If the modulo 5 counter reads 0, then a 10-bit symbol is extracted, but only eight bits of it are used. In an embodiment, the assembled 8-bit character can be retained in register 204 until transmitted to GMII interface 104. The remaining two bits are retained in register

202 for use on the next GMII clock signal. If modulo 5 counter 206 reads 1, then a 10-bit symbol is extracted, but only six bits of it are used and what is transmitted combines the two bits stored in register 202 plus these six extracted bits. This results in four bits being left over, which are stored in register 202. If modulo 5 counter 206 reads 2, then the four left-over bits from register 202 are combined with four bits from the next 10-bit symbol that is extracted from FIFO 200, resulting in six bits being left over from the extracted 10-bit symbol. These remaining six bits are retained in register 202. If modulo 5 counter 206 reads 3, then the six bits in register 202 are combined with two bits from the next extracted symbol, and the remaining eight bits are stored in register 202. If modulo 5 counter 206 reads 4, then no new symbol is extracted, and the eight left over bits are sent over GMII interface 104. If modulo 5 counter 206 has a zero value, then the 1394 side of port 114 does not push a 10-bit number into FIFO 200. PHY 102 ensures that the GMII RX clock is locked to the GMII TX clock. If no symbol is available in FIFO 200, TX Adaptation module 220 inserts an illegal or null 10-bit symbol into FIFO 200.

[0026] For translation of data in the opposite direction, where 10-bit symbols are formed from 8-bit characters, synchronization characters (i.e. illegal characters) force the value of modulo 5 counter 206 to zero and hold it there. The first valid 8-bit character that is received via connection 120 is copied into register 208 with the counter value still at zero, so the 1394 side of port 114 takes no action. The next 8-bit character received over connection 120 is placed in register 210. The 1394 side of port 114 assembles a 10-bit symbol by taking the eight bits from register 208, appending two bits from register 210, and treating the resulting10-bit symbol as if it had been received on 1394b port 108. Most such symbols are pushed into the bport RX FIFO, while others, such as repeated request or control symbols, and illegal symbols are ignored. RX Adaptation module 222 deletes the null or illegal 10-bit symbols, and uses the deletion frequency to

control the RX symbol clock phase locked loop. Thus, the RX symbol clock is phase locked to the TX symbol clock. Elasticity FIFO 224 compensates for ppm differences between the two PHY clocks. The resulting 10-bit symbol is pushed into FIFO 116, where conventional 1394b deletable symbols functionality prevents bport FIFO underflow and overflow. In an embodiment FIFO 116 is sized slightly larger than FIFO 110 to compensate for the "4inarushthenawait" symbol arrival characteristics. The remaining bits from register 210 are then stored in register 208, and the next 8-bit byte received is stored in register 210. Assembly of 10-bit symbols continues by using all of the bits from register 208, and appending any bits needed from register 210 to form the 10-bit symbol. This 10-bit symbol process continues until register 210 is emptied upon appending bits, at which point the process repeats itself by storing an 8-bit byte in both registers 208, 210 again.

[0027] The relationship between 10-bit symbols and 8-bit characters can be conceptualized as illustrated in FIG. 4. In an embodiment, port 114 can include four 10-bit registers on the 1394 side of port 114 (reference numeral 250), and five 8-bit registers on the GMII side of port 114 (reference numeral 252), as illustrated in FIG. 5. In an embodiment, symbol transmission begins with transmission of an illegal symbol to synchronize four 10-bit registers on the 1394b side of port 114 with the five 8-bit registers on the GMII side of port 114. In this manner, a 10-bit symbol transmitted immediately after an illegal symbol is received as a valid 10-bit symbol. A clock domain crossing transmit FIFO bridges the 1394b PHY clock (running at 98MHz, in an embodiment) and the 1000 base clock domains. When the clock domain crossing FIFO drains below a predetermined threshold such that it will underflow after the next five symbols are transmitted, an illegal symbol is fed into the clock domain crossing FIFO.

[0028] In an embodiment of the present invention, training and operation symbols are transmitted to support scrambler synchronization and a port synchronization handshake and can utilize K28.5 substitution. Rather than GHz logic, or phase locked loops, the present invention utilizes single clock synchronous logic in the receiver on PHY 106. Bytes are presented to port 114 from GMII interface 104 synchronously to GMII interface 104's RX clock (running at 125MHz, in an embodiment), and, unlike conventional1394b ports, there is no requirement for any high-speed bit receive logic or clock recovery in port 114. The K28.5 symbol in the IBM 8B10B code is a symbol used in normal operation that contains a special "comma" pattern sequence, denoted as 0011111 or 110000, depending on disparity. When a receiver initializes upon start up, it is unaware of where the 10-bit boundaries occur in an incoming bit stream. But when it recognizes a received K28.5 symbol (the comma sequence), it can use this symbol to determine the 10-bit boundary, and thus acquire symbol synchronization. 1394b-compliant devices perform symbol synchronization by performing K28.5 substitution, replacing D28.0 symbols with K28.5 symbols. The receiver uses the K28.5 symbol to acquire symbol synchronization, and also substitutes back the D28.0 symbol for any K28.5 symbols it finds in the stream. 1394b-compliant devices send a training symbol request as the first symbol on a port. This training symbol is scrambled, and the resulting 8B10B data symbol is transmitted. Scrambling the training symbol ensures that periodically the received data symbol is a D28.0 symbol, and the K28.5 substitution can be performed.

[0029] When port 114 has acquired symbol and scrambler synchronization, it changes the symbol it is transmitting to an operation symbol. This operation symbol conveys another request, and is scrambled onto 8B10B data symbols in just the same way. As a result of scrambling, a D28.0 symbol is still due to be transmitted occasionally, and K28.5 substitution is performed.

[0030] When port 114 is both transmitting and receiving operation request symbols, it is synchronized to its peer node, and also made aware that the peer node is synchronized to port 114. Port training is now considered complete, and port 114 is ready to transmit requests, control symbols, and data in its support of the 1394 protocols. The 5-to-4 relationship between states on the 1394b side and states on the GMII side of port 114 can be illustrated in FIG. 5. In an embodiment, port 114 utilizes five 8-bit registers (as seen on the GMII side of port 114) a, b, c, d and e, which correspond to four 10-bit registers on the 1394b side of port 114. On the GMII side of port 114, each successive 8-bit character is pushed into register a, b, c, d or e using a value read from modulo 5 counter 206 at intervals denoted by the RX clock on GMII interface 104. On the 1394b side of port 114, 10-bit data is pulled from the GMII side of port 114 using a clock that runs at 4/5 of the GMII RX clock frequency (100MHz in an embodiment). This clock increments modulo 5 counter 206 for each symbol read. The successive values read are a[7:0] || b[7:6], b[5:0] || c[7:4], c[3:0] || d[7:2] and finally d[1:0] || e[7:0].

Clocking appears as shown in Table 1.

```
GMII clock  |    |    |    |    |    |    |
write     a    b    c    d    e    a    b
100MHz clock |     |     |     |     |     |
phase offset needed <
1394b clock  |     |     |     |     |     |
read            ab     bc     cd     de     ab
```

Table 1

[0031] In an embodiment, illegal symbols are transmitted after the 0 modulo 5 symbols have been transmitted since the last illegal symbol was transmitted, and are also used to synchronize the four 10-bit registers on the 1394b side with the five 8-bit registers on the GMII side. Read and write clock pulses are aligned with the necessary phase offset for the first symbol received after a synchronizing symbol, and the read and write selectors reset to point to register "a" for write and register "ab" for read. The first 10-bit symbol is read one read clock pulse later, after values have been written to both registers a and b. While the four 10-bit registers and five 8-bit registers are synchronizing with each other, no symbols are read out, nor are any 10-bit symbols pushed into the bport RX FIFO using the 100MHz clock.

[0032] The present invention implements the 8-bit GMII interface 104 as a serial interface, by shifting a byte into the 10-bit register on the 1394b side using serial interface FIFO 200. Whenever an illegal symbol is shifted into 10-bit register, it is deleted from FIFO 200. The illegal symbol can be inserted into any stream, and deleted at the other end, even without byte/symbol synchronization.

[0033] Directing attention to FIG. 6, in another embodiment, the present invention can be implemented without RX symbol clock recovery. In this embodiment, the RX clock on PHY 102 is locked to the TX clock on PHY 102. 10-bit symbols are placed in FIFO 200 using flagged encoding. TX adaptation module 220 extracts the 10-bit symbols from FIFO 200 and places them in registers 202, 204. In this embodiment, 802.3-compliant idle bytes are used rather than inserting 10-bit null or illegal symbols. If there is no symbol available in FIFO 200, then TX Adaptation module 220 de-asserts a transmit enabled state (TX_EN) on GMII interface 104 for 11 clock periods, which causes PHY 102 to insert 802.3-compliant idle bytes. This embodiment thus meets a requirement in the 802.3 standard to periodically insert idle bytes. If there is no symbol in FIFO

200 then TX-EN is de-asserted on PHY 102 for 11 clock periods, which causes PHY 102 to insert the idle bytes. In this embodiment, FIFO 200 is appropriately sized to buffer nine 10-bit symbols that accumulate during the 11 clock periods. On the receive side, RX Adaptation module 222 only takes data from GMII interface 104 when receive data valid (RX_DV) is asserted, which is not asserted when 802.3-compliant idle bytes are being received. Using flagged decoding, 10-bit decoded symbols are pushed into the receive FIFO when available when available. Usually, this occurs on four out of every five clock cycles, but with bursts of 11 clocks with no push when RX_DV is de-asserted. In an embodiment, elasticity FIFO 224 handles both the ppm differences between PHY 106 and PHY 102, as well as the frequency difference between the 802.3 bit rate and the 1394 S800 bit rate. Elasticity FIFO 224 thus must be at least nine symbols deeper than used in 1394b implementations to compensate for the bursty nature of reception. However, in this embodiment, there is no need for PHY 102 to recover the TX symbol clock.

[0034] Directing attention to FIG. 7, in yet another embodiment, robust encoding is realized, which implements TX adaptation phase knowledge. PHY 102 ensures that the GMII RX clock is locked to the GMII TX clock. Robust encoding module 230 takes a 10-bit symbol from FIFO 200 on four out of every five GMII TX clocks, and uses TX Adaptation module 220 to determine the encoding to be used, thus generating a 10-bit encoded symbol. TX Adaptation module 220places the extracted symbol into registers 202, 204 to generate on every GMII TX clock an 8-bit byte as described above. If there is no symbol available in FIFO 200, then robust encoding module 230 instructs TX Adaptation module 220 accordingly. TX Adaptation module 220 de-asserts TX_EN on GMII interface 104 for 11 clock periods, which causes PHY 102 to insert 802.3-compliant idle bytes. FIFO 200 is appropriately dimensioned to buffer nine symbols.

[0035] The receive side in this embodiment is similar to the receive side illustrated in FIG. 6 and explained above, with the addition of robust decoding module 232. Robust decoding module 232 decodes the 10-bit symbols that result from RX Adaptation module 222's conversion to 10-bit, encoded symbols from 8-bit, encoded bytes received from PHY 102.

[0036] While numerous methods and apparatus for transmitting 1394-compliant symbols using a gigabit Ethernet PHY have been illustrated and described in detail, it is to be understood that many modifications may be made to embodiments of the present invention without departing from the spirit thereof.